Computer-Assisted Language Comparison in Practice

Tutorials on Computational Approaches to the History and Diversity of Languages

PyLexibench — Generating Data for Lexibench with a Python Package

Luise Häuser / Robert Forkel / Johann-Mattis List
Computational Molecular Evolution Group / DLCE/ Chair for Multilingual Computational Linguistics
Heidelberg Institute for Theoretical Studies / MPI-EVA Leipzig / University of Passau

With PyLexibench we introduce a small Python package that can be used to populate the Lexibench benchmark for computational historical linguistics with benchmark data. Here, we introduce the package and show how it helps to access and expand Lexibench. We also introduce new data for character matrices in various forms and formats and lay out how we intend to use the package to manage Lexibench releases in the future.

1 Introduction

A selection of suitable datasets is the basis for meaningful benchmarks. Benchmarks themselves are of great importance in the development of computational methods that address general and more specific tasks in a discipline. After recently introducing Lexibench as a collection of benchmark data for computational historical linguistics (Häuser and List 2025), we felt the need to complement the database with more regular means to access and modify the data in order to allow scholars to adjust the benchmark for targeted applications. We therefore decided to create a small Python package that would conduct this task. As a result, Lexibench is now complement with PyLexibench, a Python package that provides standardized access to data from the Lexibank repository (Blum et al. 2025).

While we provided custom scripts that would help to download and arrange the data when introducting Lexibench previously, PyLexibench now formalizes this process. After installing PyLexibench, all one needs to download and prepare Lexibench data is a valid table with links to CLDF datasets that can be used to populate individual Lexibench repositories (see § 4). In addition, we have now also extended the codebase

Computer-Assisted Language Comparison in Practice
Volume 8, Number 1
URL: https://calc.hypotheses.org/8267 DOI: 10.15475/calcip.2025.1.4
Published under a Creative Commons Attributions 4.0 LICENSE
Published on 28/04/2025

in such a way that various character matrices for Lexibench datasets can be constructed as well. These matrices — provided in common forms typically used in phylogenetic software packages — can then be directly used for all kinds of experiments (see § 3.3). In addition, PyLexibench now also allows for the direct integration of Glottolog, from which reference trees can now be retrieved to form gold standard trees that may come in handy during evaluation.

In the following, we will first give a very brief background on the kind of data that can be assembled with PyLexibench. We will then introduce in detail the basic aspects of the data and the methods that PyLexibench offers, putting a particular emphasis on the creation of character matrices in various flavors. Following this overview, we will then share a concrete example that runs interested users through the major process of creating benchmark data from collections of CLDF datasets with the help of PyLexibank. We conclude by pointing to future plans and open questions that we encountered when working towards a first official release of Lexibench and PyLexibench.

2 Background

Computational tools from phylogenetics are now commonly applied in historical linguistics. This quantitative turn (Geisler and List 2022) is reflected in the fact that almost all recent linguistic studies on phylogenetic reconstruction work with computational approaches. It has also led to a surge in new phylogenies for many of the world's larger and smaller language families (Greenhill et al. 2023, Savelyev et al. 2020, Kassian et al. 2021).

Experiments based on cognate data usually consist of a multi-stage pipeline (see Geisler and List 2010, Häuser et al. 2024, and Häuser et al. 2024b). In a first step, wordlists are extracted from the native cognate data. In the second step, the datasets are encoded as character matrices based on these word lists. As a rule, these are binary character matrices, but there are also alternative representations (see Häuser et al. 2024b), which we outline below. The character matrices are then used as input for phylogenetic inference. To score the resulting trees, these trees are compared to the gold standard from Glottolog. This requires the extraction of a reference tree suitable for the respective cognate dataset. PyLexibench covers three main steps of such a pipeline, namely the extraction of wordlists, the construction of character matrices and the extraction of reference trees.

3 Materials and Methods

3.1 CLDF Datasets from the Lexibank Repository

Main data underlying PYLe are CLDF datasets that conform to the particular requirements for CLDF wordlists to be included as part of the Lexibank repository (Blum et al. 2025). Lexibank itself offers a large wordlist along with computed phonetic and lexical features inferred from standardized wordlists, but the core idea of Lexibank is not only to share an aggregated database but more importantly to assemble individual CLDF datasets that share enough basic structures to allow for the direct comparison and integration without having to tweak the data by any additional computational means.

The core of each Lexibank version is a table that provides links to the destinations from which CLDF datasets that conform to the extended requirements for Lexibank datasets can be downloaded. These are represented via a DOI that itself links to the Zenodo archive (https://zenodo.org), as well as a link to the GIT repository (currently assuming data to be curated on GitHub), where the actual version of individual CLDF datasets can be found.

We use the basic structure of this table, which is available in the form of a TSV file, as the base from which we populate Lexibench datasets. A Lexibench dataset in this form is nothing else then a dataset that was created with the help of PyLexibench and that provides a specific TSV file that provides information on the download locations. The file that we use is the same one that we shared along with the first Lexibench dataset (Häuser and List 2025), which in turn was taken from the release of Lexibank 2 (Blum et al. 2025) and modified by deleting those datasets that we did not consider as suitable for the inclusion into a benchmark database that provides gold standard cognate sets provided by experts. In this form, it can be found in the folder etc under the filename lexibank.tsv in the official Lexibench repository (https://codeberg.org/lexibank/lexibench).

The simplicity of the format (requiring only a TSV file placed in a particular folder of a repository) should make it easy to provide new releases of Lexibench in the future, while at the same time allowing colleagues to create their own releases of Lexibench, depending on the data that they require in their studies.

3.2 The PyLexibank Software Package

PyLexibank is a Python package that is constructed in such a way that its commands can all be invoked from the command line. We are still working on the first release of the package (which would then also lead to an official release of Lexibench), but the package can be easily installed by downloading the data with GIT and then installing the package via the command line from a fresh virtual environment, as shown below (using the release 1.0 that we used for this study).

```
$ pip install pylexibench
$ lexibench --help
```

The main website of the project (curated on Codeberg, https://codeberg.org/lexibank/pylexibench) provides detailed instructions on how to run the package in order to download and prepare several different kinds of datasets for phylogenetic analysis in comparative linguistics.

3.3 Construction of Character Matrices

As a feature we newly introduced, we now construct different kinds of character matrices in Lexibench. These are derived from the annotated cognate sets and can be directly loaded from various software tools for phylogenetic reconstruction.

A cognate dataset for a given concept list and a set of languages can be understood as an assignment of a set of words to each language-concept pair. Words that share a common ancestor are grouped together in cognate sets (List 2016). From these groupings, we generate a matrix that assigns a set of cognate sets to each language-concept pair. It serves as the basis for the construction of the different character matrix types that can later be used in phylogenetic reconstruction. We usually assume that the concept lists are well-constructed, meaning that each language has at least one word for every concept. If there is no word provided for a particular language-concept pair in the original data, we treat it as missing information. If there is more than one word provided for a language-concept pair, we call this a polymorphism. Table 1 shows the structure of a typical cognate dataset, consisting of the triple of Language, Concept, and (Word) Form, to which cognate sets are assigned.

Language	**Concept**	**Form**	**Cognate Set**
English	BIG	big	big_1
English	BIG	great	big_2
German	BIG	groß	big_2
Dutch	BIG	groot	big_2
Norwegian	BIG	stor	big_3
Swedish	BIG	stor	big_3

Table 1: Basic structure of typical datasets.

A cognate dataset can be represented by a binary character matrix using the symbols 0 and 1. We obtain it as the presence-absence-matrix corresponding to the matrix containing the cognate sets. In a binary character matrix, each concept is represented by as many columns as there are cognate sets with each column corresponding to a specific cognate set. If a language has a word belonging to a particular cognate set, the corresponding entry is set to 1. Otherwise, it is set to 0. We assume that for each concept there exists at least one word in every language. If no cognate set is provided for a language-concept pair, this is treated as missing information, and all corresponding columns for that concept are marked by a specific character indicating missing data (normally a dash in computational approaches). The tabular information can be stored in different formats that can then be used as input for phylogenetic reconstruction methods, typically designed for the application in bioinformatics. Among these formats, PyLexibench by now supports export to the general Nexus format (Maddison et al. 1997) that can be used in many different software packages, and the so-called Relaxed-Phylip format, which was originally designed for the Phylip software package (Felsenstein 2021) but is now supported by many more software tools. Table 2 shows the binary character matrix corresponding to the small cognate dataset in Table 1.

Language	big_1	big_2	big_3
English	1	1	0
German	0	1	0
Dutch	0	1	0
Norwegian	0	0	1
Swedish	0	0	1

Table 2: Binary character matrix corresponding to the cognate dataset in Table 1.

Cognate datasets can also be represented by a multi-state character matrix. The binary matrix representation splits the meaning slot into as many cognate sets as one can observe. It models the evolution of the cognate sets as a process of cognate gain and cognate loss that proceeds independently of the original meaning that the cognate words express in the languages under consideration. The basic unit in which evolution happens in a multi-state model is the concept itself. Here, language change is no longer modeled as a process in which new words are gained and old words are lost, but rather as a process in which a given meaning is expressed by different word forms that may alternate during language change (see List 2016 for a closer discussion of binary state representations compared to multi-state representations in phylogenetic approaches in historical linguistics).

A considerable draw-back of multi-state representations is not trivial to model those cases where we find two word forms for one concept in the same language. These cases of synonymy, leading to polymorphisms in phylogenetic datasets, are usually excluded when modeling the evolution of cognate sets in multi-state approaches. However, some approaches allow to handle them.

Another limitation of multi-state approaches is that software created for applications in biology usually works with a limited number of different character states — that is, cognate sets per concept. Thus, the MULTISTATE alphabet used in RAxML-NG (Kozlov et al. 2019), for example, only offers 64 different symbols. While this may be enough to model cognate sets with multi-state representations for several dozen languages (64, if we assume that polymorphisms would have been actively resolved beforehand), it would quickly become insufficient when working with language families consisting of more than a hundred languages, such as Indo-European or Austronesian.

Nevertheless, since there is a sufficient number of datasets in Lexibench that qualify for a multi-state representation, PyLexibench allows to create multi-state character matrices from cognate datasets that follow the traditional Phylip format for the representation of multiple sequence alignments.

The character matrices described so far are all in a sense deterministic, as we assume that a fixed symbol is observed at each column for each language. In contrast, a probabilistic character matrix allows for multiple symbols to occur with specific probabilities, which are explicitly provided in the matrix. To represent missing data we explicitly set the probabilities for all symbols to 1.0 (Kozlov et al. 2019), which is an encoding that does not provide any information.

Probabilistic character matrices can be leveraged for cognate data by interpreting the datasets in a probabilistic manner. If there are several synonyms for a concept in a language, it can be simplified to assume that each of these synonyms occurs with the same probability. Using this approach, we can construct both probabilistic binary and multistate character matrices.

In a probabilistic binary state character matrix, each concept is represented by as many columns as there are cognate sets, just like in its deterministic counterpart. Let us consider the cognate set of one of the k synonyms existing for a particular language-concept pair. At the corresponding entry in the matrix we observe the symbol 1 with the probability 1/k and the symbol 0 with the probability 1 – 1/k. In the case of English big and great, as used in the previous examples, k would be 2 and the probabilities that we must fill in for the cognate sets big_1 and big_2 are 0.5 for 1 and 0, respectively, in both cognate sets. We represent these probabilities with the help of tuples of two numbers, the first number represents the probability that the character is not present and therefore corresponding to a 0 in a non-probabilistic binary matrix, while the second number represents the probability of observing a character, corresponding to a 1, respectively. Table 3 illustrates this coding for our little toy sample of Germanic words for “big”.

Cognate Set	**big_1**	**big_1**	**big_2**	**big_2**	**big_3**	**big_3**
States	0	1	0	1	0	1
English	0.5	0.5	0.5	0.5	1.0	0.0
German	1.0	0.0	0.0	1.0	1.0	0.0
Danish	1.0	0.0	0.0	1.0	1.0	0.0
Norwegian	1.0	0.0	1.0	0.0	0.0	1.0
Swedish	1.0	0.0	1.0	0.0	0.0	1.0

Table 3: Probabilistic binary character matrix where cognate sets represent the basic unit of evolution and character states can be 0 or 1.

In a probabilistic multi-state character matrix, the basic character is the concept and cognate sets correspond to the different states that the character can take. Probabilities for each state are again represented by floating point numbers, and provided for each state. Polymorphisms are handled in the same way as they are in probabilistic binary state character matrices, with the exception that each cognate set is represented by one number between 0 (character state is not active) and 1 (character state is active). Synonyms can again be easily handled by assigning the probability 1/k to each state.

Concept	**big**	**big**	**big**
States	big_1	big_2	big_3
English	0.5	0.5	0.0
German	0.0	1.0	0.0
Danish	0.0	1.0	0.0
Norwegian	0.0	0.0	1.0
Swedish	0.0	0.0	1.0

Table 4: Probabilistic multi-state character matrix where concepts represent the basic units and cognate sets represent character states.

To write probabilistic character matrices to files, PyLexibench uses the dedicated CATG format introduced by Kozlov et al. (2019). RAxML-NG fully supports this format in its tree inferences (details can be found in Häuser et al. 2024b).

For each dataset, PyLexibench extracts two sets of character matrices. While for the first set, all avaible data is taken into account, the second set of character matrices is selected in such a way that it is fully compatible with Glottolog’s phylogenies. This means that it only contains data for languages, for which Glottocodes have been provided. As a result, character matrices in the second set can be smaller compared to those in the first set regarding the number of languages. The advantage of these smaller datasets is, however, that any the trees inferred from compatible character matrices can be compared to the Glottolog classification and thus use it as a rudimentary gold standard to evaluate the accuracy of phylogenetic inference approaches (Rama et al. 2018).

3.4 Using Glottolog's Classification as a Gold Standard

For those who wish to use Glottolog's classifications of the languages of the world as a gold standard that could be used to help evaluate or improve methods for phylogenetic classification, trees from Glottolog are automatically added for derived datasets that are pruned in such a way that all languages referenced by a valid Glottocode in the original CLDF data are used to extract phylogenetic trees from Glottolog via the PyGlottolog API (https://pypi.org/project/pyglottolog). The Glottolog tree for a particular derived dataset is the sub-tree of the full classification from Glottolog comprising exactly the same Glottocodes (and hence also the same languages) as the derived dataset itself. This underlines the advantage of providing consistent Glottocodes for all varieties referenced in particular in CLDF datasets, given that this allows us to retrieve the information on the phylogenies underlying families in individual samples directly, without having to resort to any form of manual checking and intervention of this data.

3.5 Dataset Properties

With PyLexibench, we extend the statistics provided in Lexibench by additional features. The features themselves are related to different kinds of the benchmark, with some being computed from the multilingual wordlists, and other properties being computed from the character matrices. As basic wordlist properties, we list the language family of the *languages* being compared, the *number of concepts*, *languages*, and *words*, as well as the *average mutual coverage* (referring to the percentage of overlap of individual wordlists in a dataset, introduced in List et al. 2018), and the *cognate diversity* of a concept list (comparing the number of cognate sets and the number of words, introduced in List 2014).

For character matrices, we now compute the *sizes of cognate sets per concept* (referred to as charset sizes in the context of bioinformatics), reporting the *maximum*, *minimum*, and *median* values, as well as the *standard deviation* for each dataset. This score, that has not been considered much in linguistic analyses so far, is particularly useful in detecting coding problems. If, for example, a dataset has an extreme amount of synonyms — resulting, for example, from an erroneous merger of distinct concepts –, this may easily end up resulting in extremely large amounts of cognate sets per concept, exceeding the number of languages in a dataset. We also report the *polymorphic cell ratio* and the *polymorphic concept ratio*. The former refers to the ratio between translational equivalents with synonyms (language-concept pairs with more than one word form) and entries with only one translation. The latter refers to the ratio of concepts that exhibit at least one case of synonymy and concepts without synonymy in a wordlist. Finally, we report the average number of concepts that have a translational equivalent per language variety.

3.6 Implementation and Release Plans

With PyLexibench, we hope to be able to faciliate new releases of Lexibench in the future. The basic strategy for releasing Lexibench datasets is to install the most recent version of PyLexibench and then run the basic commands required for a new release in the command line. The advantage of this practice is that users who wish to release their own version of Lexibench — using only a selection of the data or providing even more data than in official releases — can use the workflow along with the PyLexibank package to either publish their own releases or to supplement the data accompanying their studies by providing the datasets along with the commands in PyLexibench that they used to create them. In this way, benchmarks can be targeted to particular studies instead of having to filter particular parts of the officially released data.

PyLexibench itself is currently available as version 1.0, now yielding version 1.0 of Lexibench. PyLexibench is curated on Codeberg (https://codeberg.org/lexibank/pylexibench) and archived with the Python Package Index (https://pypi.org/project/pylexibench). Lexibench is also curated on Codeberg (https://codeberg.org/lexibank/lexibench) and archived with Zenodo (https://doi.org/10.5281/zenodo.15260463).

4 Populating Lexibench using PyLexibench

In order to populate our Lexibench benchmark database using PyLexibench, all that needs to be done is to (1) install the PyLexibenk package from the Python Package Index, using a fresh virtual environment, (2) create a folder in which one has to place the file that contains the information about the datasets that one wants to include in a given release, and (3) download the most recent Glottolog data via GIT. These steps can all be easily carried out in the commandline, as shown below.

```
$ pip install pylexibench
$ mkdir lexibench
$ mkdir lexibench/etc/
$ wget https://codeberg.org/lexibank/lexibench/raw/
  branch/main/etc/lexibank.tsv
  -O lexibench/etc/lexibank.tsv
$ git clone https://github.com/glottolog/glottolog.git
  --depth 1 --branch v5.1
```

The clone of the Glottolog data in this example only fetches the most recent release. This has the advantage that the download saves a lot of space, since we ignore all the history.

After these steps have been carried out, it is easy to download the data and to create the wordlists as a first step of the benchmark creation process. The installation of PyLexibench creates the command lexibench on the commandline. This command itself runs via subcommands that carry out different tasks, as shown below.

```
$ lexibench --repos=lexibench download
$ lexibench --repos=lexibench lingpy_wordlists
  --language-threshold=4 --concept-threshold=85
  --coverage-threshold=0.45
```

The first command downloads the original CLDF datasets to the folder downloads in the lexibench folder, adding some statistics. The second command creates LingPy wordlists that can be easily processed by the LingPy library (that is taken as the basis for later conversions).

Based on these wordlists, we can now test two basic methods for automated cognate detection (as we have also done when presenting the Lexibench benchmark the first time in Häuser and List 2025), the SCA approach (first presented in List 2012), and the LexStat approach (also first presented in List 2012, later refined in List 2017). While SCA can be applied to all wordlists that we sample for Lexibench, LexStat is better limited to datasets with no more than 50 languages. This is handled by passing the respective keyword parameters in the command line, as shown below for both commands. The results will be written to a folder `lingpy_cognates` along with statistics that show evaluation scores for both methods. Offering the results of these tests in this form has the advantage that future tests with novel approaches to cognate detection can be done without having to rerun the tests with the SCA and the LexStat baseline, as long as the same data and evaluation methods are being used. When running the subcommand with the option `lexstat`, SCA cognates will also be calculated and compared directly to the LexStat cognates (on the sample of datasets with no more than 50 varieties). The results of the command show that LexStat outperforms SCA by two points in the B-Cubed F-Scores, confirming previous studies (List et al. 2017).

```
$ lexibench --repos=lexibench lingpy_cognates lexstat
  --lexstat-threshold=0.55 --language-threshold=50
```

An additional subcommand extracts all base phylogenies from Glottolog, based on the assignment of Glottocodes in the individual CLDF datasets. The resulting phylogenies are written to files in Newick format in the folder glottolog_trees and can be compared to automatically inferred phylogenies or to serve as a backbone for additional methods, such as, for example, ancestral state reconstruction (List 2016, Jäger and List 2018), or tree-based borrowing detection methods (Nelson-Sathi et al. 2011, List et al. 2014). Since not all languages in the original CLDF data are linked to Glottolog, the command will also prune the data, retaining only those languages that are linked to valid Glottocodes.

```
$ lexibench --repos=lexibench glottolog_trees
  --glottolog=glottolog
```

Having extracted the reference trees from Glottolog, character matrices can be written to file. When running the subcommand, binary state and multi-state matrices — both deterministic and stochastic — in standard formats used in bioinformatics will be produced in two forms, one reflecting the pruned form of the data where all languages are linked to valid Glottocodes, and one reflecting all languages. Multi-state matrices are restricted to those cases where the number of cognate sets per concept does not exceed the maximum number of characters in the basic formats used in bioinformatics (as mentioned above). The two datasets are written to two different folders, with the folder character_matrices being used for data containing all language varieties and the folder character_matrices_compatible being used to store the matrices for datasets that have been pruned to reflect the Glottolog phylogenies. The command allows to specify a given Glottolog version. Since we have, however, downloaded only the most recent version, the specification is not necessary in the command that we use to illustrate the workflow.

```
$ lexibench --repos=lexibench character_matrices
  --glottolog=glottolog
```

Having run all these commands accordingly, we are left with a repository that is filled with several files, including wordlists, phylogenetic trees, and character matrices. This repository has now also been released as version 1.0 of Lexibench on Codeberg (https://codeberg.org/lexibank/lexibench.git) and archived with Zenodo (https://doi.org/10.5281/zenodo.15260463).

5 Conclusion

In this study, we have introduced PyLexibench, a small Python package that provides a more principled and extended version of the code that we first wrote in order to populate benchmark data for computational historical linguistics as part of Lexibench. While there are still many aspects of the package and the benchmark database itself that remain to be tested more rigorously in the future, we consider the data and the code that generates the data as useful enough at this point to share them directly in an official first version. This does not mean that we consider that the code and the data that the code produces are free from errors. However, we consider them interesting and hopefully also useful enough to be tested and applied by colleagues working in phylogenetic reconstruction and other computational approaches in the field of comparative linguistics.

References

Blum, Frederic, Carlos Barrientos, Johannes Englisch, Robert Forkel, Russell D. Gray, Simon J. Greenhill, Christoph Rzymski, and Johann-Mattis List. 2025. Lexibank²: Precomputed Features for Large-Scale Lexical Data. Geneva: Zenodo. https://doi.org/10.5281/zenodo.14800315

Felsenstein, J. 2021. “PHYLIP: Phylogeny Inference Package) [Software, Version 3.698].” https://phylipweb.github.io/phylip/

Forkel, Robert, Johann-Mattis List, Christoph Rzymski, and Guillaume Segerer. 2024. “Linguistic Survey of India and Polyglotta Africana: Two Retrostandardized Digital Editions of Large Historical Collections of Multilingual Wordlists.” In Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), edited by Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue, 10578–83. Torino, Italy: ELRA; ICCL. https://aclanthology.org/2024.lrec-main.925

Geisler, Hans J., and Johann-Mattis List. 2022. “Of Word Families and Language Trees: New and Old Metaphors in Studies on Language History.” Moderna 24 (1-2): 134–48. https://doi.org/10.19272/202201902005

Geisler, Hans, and Johann-Mattis List. 2010. “Beautiful Trees on Unstable Ground. Notes on the Data Problem in Lexicostatistics.” In Die Ausbreitung Des Indogermanischen. Thesen Aus Sprachwissenschaft, Archäologie Und Genetik, edited by Heinrich Hettrich. Wiesbaden: Reichert. [preprint, article was never published since volume was canceled] https://hal.archives-ouvertes.fr/hal-01298493

Greenhill, Simon J., Hannah J. Haynie, Robert M. Ross, Angela Chira, Johann-Mattis List, Lyle Campbell, Carlos A. Botero, and Russell D. Gray. 2023. “A Recent Northern Origin for the Uto-Aztecan Family.” Language 99 (1). 81-107. https://doi.org/10.1353/lan.2023.0006

Hammarström, Harald, Martin Haspelmath, Robert Forkel, and Sebastian Bank. 2024. Glottolog [Dataset, Version 5.1]. Leipzig: Max Planck Institute for Evolutionary Anthropology. https://glottolog.org

Häuser, Luise, and Johann-Mattis List. 2025. “Lexibench: Towards an Improved Collection of Benchmark Data for Computational Historical Linguistics.” Computer-Assisted Language Comparison in Practice 8: 9–16. https://doi.org/10.15475/CALCIP.2025.1.2

Jäger, Gerhard, and Johann-Mattis List. 2018. “Using Ancestral State Reconstruction Methods for Onomasiological Reconstruction in Multilingual Word Lists.” Language Dynamics and Change 8 (1): 22–54. https://doi.org/10.1163/22105832-00801002

Kassian, Alexei S., Mikhail Zhivlov, George Starostin, Artem A. Trofimov, Petr A. Kocharov, Anna Kuritsyna, and Mikhail N. Saenko. 2021. “Rapid Radiation of the Inner Indo-European Languages: An Advanced Approach to Indo-European Lexicostatistics.” Linguistics 59 (4): 949–79. https://doi.org/10.1515/ling-2020-0060

Kozlov, Alexey M, Diego Darriba, Tomáš Flouri, Benoit Morel, and Alexandros Stamatakis. 2019. “RAxML-NG: A Fast, Scalable and User-Friendly Tool for Maximum Likelihood Phylogenetic Inference.” Edited by Jonathan Wren. Bioinformatics 35 (21): 4453–55. https://doi.org/10.1093/bioinformatics/btz305

List, Johann-Mattis. 2012. “LexStat. Automatic Detection of Cognates in Multilingual Wordlists.” In Proceedings of the EACL 2012 Joint Workshop of Visualization of Linguistic Patterns and Uncovering Language History from Multilingual Resources, 117–25. Stroudsburg. https://aclanthology.org/W12-0216/

List, Johann-Mattis. 2014. Sequence Comparison in Historical Linguistics. Düsseldorf: Düsseldorf University Press. https://sequencecomparison.github.io

List, Johann-Mattis. 2016. “Beyond Cognacy: Historical Relations Between Words and Their Implication for Phylogenetic Reconstruction.” Journal of Language Evolution 1 (2): 119–36. https://doi.org/10.1093/jole/lzw006

List, Johann-Mattis, Simon J. Greenhill, Cormac Anderson, Thomas Mayer, Tiago Tresoldi, and Robert Forkel. 2018. “CLICS². An Improved Database of Cross-Linguistic Colexifications Assembling Lexical Data with Help of Cross-Linguistic Data Formats.” Linguistic Typology 22 (2): 277–306. https://doi.org/10.1515/lingty-2018-0010

List, Johann-Mattis, Simon J. Greenhill, and Russell D. Gray. 2017. “The Potential of Automatic Word Comparison for Historical Linguistics.” PLOS ONE 12 (1): 1–18. https://doi.org/10.1371/journal.pone.0170046

List, Johann-Mattis, Shijulal Nelson-Sathi, Hans Geisler, and William Martin. 2014. “Networks of Lexical Borrowing and Lateral Gene Transfer in Language and Genome Evolution.” Bioessays 36 (2): 141–50. https://doi.org/10.1002/bies.201300096

Maddison, D. R., D. L. Swofford, and W. P. Maddison. 1997. “NEXUS: An Extensible File Format for Systematic Information.” Syst. Biol. 46 (4): 590–621.

Nelson-Sathi, Shijulal, Johann-Mattis List, Hans Geisler, Heiner Fangerau, Russell D. Gray, William Martin, and Tal Dagan. 2011. “Networks Uncover Hidden Lexical Borrowing in Indo-European Language Evolution.” Proceedings of the Royal Society of London B: Biological Sciences 278 (1713): 1794–1803. https://doi.org/10.1098/rspb.2010.1917

Rama, Taraka, Johann-Mattis List, Johannes Wahle, and Gerhard Jäger. 2018. “Are Automatic Methods for Cognate Detection Good Enough for Phylogenetic Reconstruction in Historical Linguistics?” In Proceedings of the North American Chapter of the Association of Computational Linguistics, 393–400. https://aclweb.org/anthology/N18-2063

Ronquist, Fredrik, Maxim Teslenko, Paul van der Mark, Daniel L. Ayres, Aaron Darling, Sebastian Höhna, Bret Larget, Liang Liu, Marc A. Suchard, and John P. Huelsenbeck. 2012. “MrBayes 3.2: Efficient Bayesian Phylogenetic Inference and Model Choice Across a Large Model Space.” Systematic Biology 61 (3): 539–42. https://doi.org/10.1093/sysbio/sys029

Savelyev, Alexander, and Martine Robbeets. 2020. “Bayesian Phylolinguistics Infers the Internal Structure and the Time-Depth of the Turkic Language Family.” Journal of Language Evolution 5 (1): 39–53. https://doi.org/10.1093/jole/lzz010

Supplementary Material
The code is curated on Codeberg (https://codeberg.org/lexibank/pylexibench.git) and archived with PyPi (https://pypi.org/project/pylexibench, Version 1.0). The Lexibench data are curated on Codeberg (https://codeberg.org/lexibank/lexibench.git) and archived with Zenodo (https://doi.org/10.5281/zenodo.15260463, Version 1.0).

Funding Information

This project has received funding from the European Research Council (ERC) under the European Union's Horizon Europe research and innovation programme (Grant agreement No. 101044282, JML, and Grant agreement No. 101087081, LH). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.